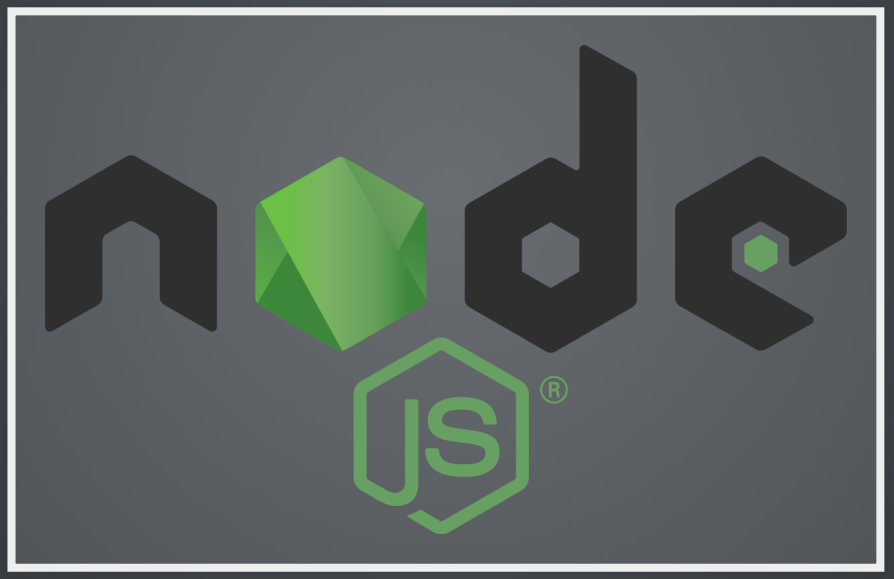
NODE.JS DÉBUTANT



1. INTRODUCTION

QU'EST-CE QUE C'EST ?

- Environnement d'exécution Javascript construit sur le moteur JavaScript V8 de Chrome.
- Environnement bas niveau permettant l’exécution de JavaScript côté serveur.

AVANTAGES

- Basé sur l’événementiel et des entrées/sorties non bloquantes, ce qui le rend léger et efficace.
- Ultra-rapide !
- Le plus grand écosystème (npm) de bibliothèques open source au monde !
- Coûts de production très bas par rapport aux technologies concurrentes.

INCONVÉNIENTS

- Fragilité du langage Javascript et de la programmation fonctionnelle vs la programmation orientée objet.
- Grosse exposition aux hackers.
- Le bon côtoie le moins bon.

HISTORIQUE

Node.js a été créé par Ryan Dahl en 2009, qui a eu cette idée après avoir observé la barre de progression d'un chargement de fichier : le navigateur ne savait pas quel pourcentage du fichier était chargé, et devait interroger le serveur web. Dahl voulait développer une méthode plus simple et quand le moteur V8 fut diffusé, il commença à s'intéresser à JS : node.js était né !

Wikipédia - https://fr.wikipedia.org/wiki/Node.js

ENVIRONNEMENT

- OS Linux Ubuntu 16.04
- Visual Studio Code
- Terminal + ZSH

INSTALLATION SOUS LINUX UBUNTU 16.04

```
$ sudo apt-get update
$ sudo apt-get install nodejs
$ nodejs -v
```

FONCTIONNEMENT

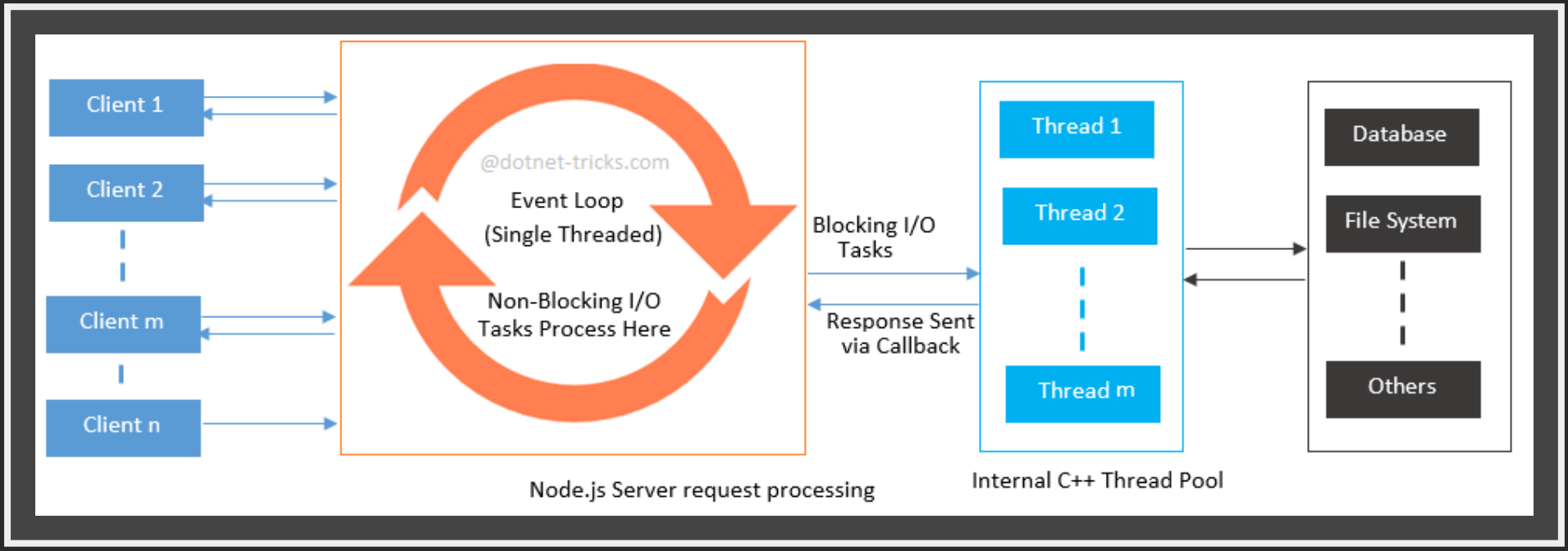
Philosophie différente des autres types de serveurs (Apache, Nginx, IIS, ...)

Un seul et unique thread

Et pourtant, c'est ultra-rapide ...

Comment est-ce possible ?

Grâce à la gestion des **EVENEMENTS** !



@ dotnettricks

Conséquences

- Encore une fois ... c'est rapide !
- Pas possible d'écrire du code procédural.
- Il faudra écrire des traitements les plus courts possibles, pour ne pas bloquer le thread principal.

Les programmes node.js seront écrits en JavaScript, et exécutés via la commande node

```
$ node app.js
```

2. PREMIERS PAS

HELLO WORLD

/demos/02-initiation/01-hello-world

```
console.log('Hello World !');
```

```
$ node index.js
```

VARIABLES LOCALES

- Variables scopées
- Scope limité au module (fichier)

/demos/02-initiation/02-local-variables

```
let myString = 'I\'m a String';
console.log(myString); // I'm a String
```

```
$ node index.js
```

VARIABLES GLOBALES

- Objet global embarque des objets natifs utiles
- Scope étendu au contexte d'exécution

/demos/02-initiation/03-global-variables

```
console.log(global);
```

Possible d'hydrater l'objet global avec des variables personnalisées

/demos/02-initiation/03-global-variables

```
global.myVariable = 'Toto';
global.myFunction = function(name) {
  if(!name) return false;
  console.log(`Hello ${name}`);
};
console.log(global.myVariable);
global.myFunction('Marcel');
```

```
$ node index.js
```

A noter : il n'existe pas d'objet window dans l'environnement node

ARGUMENTS

Il est possible de récupérer les arguments qui sont passés au script lors de son démarrage.

process.argv

/demos/02-initiation/04-arguments

```
function collectName() {
  if(typeof(process.argv[2]) === 'undefined')
    return 'Pas de nom trouvé';
  else
    return `Hello ${process.argv[2]}`;
}
const output = collectName();
console.log(output);
```

```
$ node index.js Sophie // Hello Sophie
$ node index.js // Pas de nom trouvé
```

UTILISER LES ENTRÉES/SORTIES

Il est possible de manipuler les entrées/sorties du terminal.

process.stdin / process.stdout

/demos/02-initiation/05-input-output

```
process.stdout.write('Hello World\n');
```

```
$ node index.js
```

/demos/02-initiation/05-input-output

```
process.stdin.on('data', function(data) {
  process.stdout.write('Data received\n');
  process.stdout.write(data);
});
```

```
$ node index.js
```

GÉRER LE TIMING

/demos/02-initiation/06-timing

```
setTimeout(function() {
  console.log('Loading ...');
}, 2000);
setTimeout(function() {
  console.log('Hello Technocité');
}, 5000);
```

```
$ node index.js
```

EXERCICES

A l'aide de process.stdout et process.stdin, écrire un programme qui va poser trois questions à l'utilisateur, enregistrer ses réponses et, une fois le process terminé, va afficher le récapitulatif et un indice calculé, au choix.

Version procédurale

/exos/02-initiation/01-ask-manager/index.js

```
const questions = [
  'What\'s your name ? \n',
  'What\'s your favorite animal ? \n',
  'What\'s your favorite color ? \n'
  ];

const responses = [];

let cursor = 0;

const ask = function() {
  process.stdout.write(questions[cursor]);
};

const resume = function() {
  let loopCycle = questions.length;
```

```
$ node index.js
```

Version "objet" en JS old school

/exos/02-initiation/01-ask-manager/index-better.js

```
const AskManager = function() {

  let responses = [], cursor = 0;

  const questions = [
    'What\'s your name ? \n',
    'What\'s your favorite animal ? \n',
    'What\'s your favorite color ? \n'
  ];

  const init = function() {
    listen();
    ask();
  };

  const ask = function() {
```

```
$ node index.js
```

3. GESTION DES MODULES

LES MODULES NATIFS

- Clairement plus orientés bas niveau
- Font partie du coeur node.js
- Sont accessibles globalement

RTFM

https://nodejs.org/api/

La caverne d'Ali Baba du développeur node.js

Une instruction javascript pour les utiliser :

```
require()
```

/demos/03-modules/01-native/

```
const util = require('util');

async function fn() {
  return 'hello world';
}
const callbackFunction = util.callbackify(fn);

callbackFunction((err, ret) => {
  if (err) throw err;
  console.log(ret);
});
```

```
$ node index.js
```

EXERCICE

Pour la Belgique, la France et l'Allemagne, vous devez renvoyer en sortie une string formatée de la manière suivante : "En {{COUNTRY}}, le taux de TVA est de {{RATE}}%".

/exos/03-modules/01-native/

```
const Util = require('util');
const data = [
  { country : 'Belgique', rate : 21 },
  { country : 'France', rate : 20 },
  { country : 'Allemagne', rate : 19 },
];
data.forEach( (item) => {
  console.log(Util.format('En %s, le taux de TVA est de %f%', ite
});
```

```
$ node index.js
```

LES PAQUETS NODE.JS

- Bas niveau et features
- Accessibles globalement après installation
- Quasiment tout ce que vous voulez a déjà été fait
- Chargés via le Node Package Manager (npm)

NODE PACKAGE MANAGER

Fondamental dans l'environnement node.js

Installation

```
$ sudo apt-get install npm
$ npm -v
$ npm -h
```

Installer un paquet localement avec npm

```
$ npm install mon-bo-paquet --save
```

Installer un paquet globalement avec npm

```
$ npm install mon-bo-paquet-global -g
```

RTFM. Encore.

https://docs.npmjs.com/

/demos/03-modules/02-package/

```
$ npm install colors --save-dev
```

```
require('colors');
try {
  let test = 'String to test';
  test.relace('test', 'break');
}
catch(e) { console.log(e.message.red); };
```

EXERCICE

Utiliser le module keyword-extractor pour recenser les occurrences de mots dans un contenu textuel.

/exos/03-modules/02-package/

```
$ npm install keyword-extractor --save-dev
```

```
const KeywordsExtractor = require('keyword-extractor');
try {
  const text = 'This is an string wich contains some words. Each
  const extract = KeywordsExtractor.extract(text, {
    language: 'english',
    remove_duplicates: false
  });
  console.log(extract);
}
catch(e) { e.message.red };
```

LE FICHIER PACKAGE.JSON

- Crucial pour le versionnement et la maintenabilité
- Contient toutes les dépendances du projet
- Généré dès l'initialisation
- Mis à jour en permanence

```
$ npm init
```

```
{
  "name": "package-json-sample",
  "version": "1.0.0",
  "description": "Basis package.json sample",
  "main": "app.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "Steve Lebleu",
  "keywords": "package, package.json, repository",
  "license": "MIT"
}
```

Le fichier peut être édité manuellement, et sera mis à jour lorsque :

- Un package sera installé en tant que dépendance directe : --save
- Un package sera installé en tant que dépendance de développement : --save-dev

MODULES CUSTOM

Il est parfois nécessaire/utile/sympa de développer un module !

/demos/03-modules/03-custom/modules/full-module-simple.js

```
module.exports = () => {
  console.log('Hello, i\'m a module !');
};
```

/demos/03-modules/03-custom/index.js

```
const moduleWhoSayHello = require('./modules/full-module');
moduleWhoSayHello(); // Hello, i'm a module !
```

/demos/03-modules/03-custom/modules/full-module-with-param.js

```
module.exports = (name) => {
  console.log(`Hello ${name}, i'm a module !`);
};
```

/demos/03-modules/03-custom/index.js

```
const moduleWhoSayHelloToMe = require('./modules/full-module-with
moduleWhoSayHelloToMe('Steve'); // Hello Steve, i'm a module !
```

/demos/03-modules/03-custom/modules/multiple-exports.js

```
exports.yourFunctionOne = () => {
  console.log('Hello, i am the first function in a module !');
}
exports.yourFunctionTwo = (parameter) => {
  console.log('Hello, i am the second function in a module, and i
  console.log(`The parameter value is : ${parameter}`);
}
```

/demos/03-modules/03-custom/index.js

```
const customUtilModule = require('./modules/multiple-exports');
customUtilModule.yourFunctionOne(); // Hello, i am the first func
customUtilModule.yourFunctionTwo(123); // The parameter value is
```

EXERCICE

Reprendre le module JS AskManager (/exos/02-initiation/01-ask-manager) et en faire un vrai module node.

/exos/03-modules/03-custom/ask.js

```
module.exports = function() {

  let responses = [], cursor = 0;

  const questions = [
    'What\'s your name ? \n',
    'What\'s your favorite animal ? \n',
    'What\'s your favorite color ? \n'
  ];

  const init = function() {
    listen();
    ask();
  };

  const ask = function() {
```

/exos/03-modules/03-custom/index.js

```
const AskManager = require(__dirname + '/modules/ask')();
AskManager.init();
```

```
$ node index.js
```

EXERCICE RÉCAPITULATIF

Vous créez un module chargé d'afficher une liste de contacts. La data source est un simple objet json. Vous devez utiliser le module Util pour formater l'âge.

Données à afficher :

- Name
- %i years
- Is active - Is not active
- Email
- Email well formated (true|false) -> helper

/exos/03-modules/04-all/helpers/isValid

```
exports.email = async (email) => {
  let regex = /^(([^<>()\[\]\\.,;:\s@"]+(\.[^<>()\[\]\\.,;:\s@"]+
  return regex.test(String(email).toLowerCase());
};
```

/exos/03-modules/04-all/modules/contact

```
const fs = require('fs');
const Util = require('util');
const IsValid = require('./../helpers/isValid');

const display = async () => {
  const contacts = JSON.parse({{JSON_DATA_SOURCE}});
  contacts.forEach( async (contact) => {
    let isWellFormatedEmail = await IsValid.email(contact.email);
    process.stdout.write(`
      ${contact.name} - ${contact.isActive ? 'Is active' : 'Is no
  });
};

exports.display = display;
```

/exos/03-modules/04-all/app.js

```
const Contact = require('./modules/contact');
Contact.display();
```

```
$ node app.js
```

4. GESTION DES ÉVÉNEMENTS

JavaScript intègre nativement la gestion des évènements

```
const element = document.getElementById('my-element');
element.addEventListener('click', function(e) {
  alert('Hello alert');
}, true);
```

Ecouter un événement

/demos/04-events/01-listener/index.js

```
const listen = function() {
  process.stdout.write('We listen the terminal\n');
  process.stdin.on('data', function(data) {
    console.log(data);
  });
};
listen();
```

Manipuler les événements

https://nodejs.org/api/events.html

/demos/04-events/02-emitter/index.js

```
const Events = require('events');
const game = new Events.EventEmitter();
game.on('success', function(message){
    console.log(message);
});
game.emit('success', 'You win');
```

EXERCICE

Vous parsez et affichez votre liste de contacts.

- Ecouter les événements 'end' et 'error'.
- Déclencher 'end' lorsque tous les contacts ont été affichés. Le déclenchement de 'end' provoque l'affichage d'un message au choix.
- Déclencher 'error' lorsqu'une erreur est rencontrée (ici, quand un contact n'a pas d'adresse email). Le déclenchement de 'end' provoque l'affichage d'un message au choix.

/exos/04-events/02-all/helpers/isValid

```
exports.email = async (email) => {
  let regex = /^(([^<>()\[\]\\.,;:\s@"]+(\.[^<>()\[\]\\.,;:\s@"]+
  return regex.test(String(email).toLowerCase());
};
```

/exos/04-events/02-all/modules/contact

```
const fs = require('fs');
const Util = require('util');
const IsValid = require('./../helpers/isValid');
const ContactEventEmitter = new Events.EventEmitter();

ContactEventEmitter.on('end', (message) => {
  console.info(message);
});

ContactEventEmitter.on('error', (message) => {
  console.error(message);
});

const display = async () => {
  const contacts = JSON.parse({{JSON_DATA_SOURCE}});
  contacts.forEach( async (contact) => {
```

/exos/04-events/02-all/app.js

```
const Contact = require('./modules/contact');
Contact.display();
```

```
$ node app.js
```

5. GESTION DES PROCESSUS

Node permet d'éxécuter des processus système natifs

/demos/05-processus/index.js

```
const exec = require('child_process').exec;
exec('node -v', function( error, stdout) {
  if(error) { throw error };
  console.log('Command executed');
  console.log(stdout);
});
```

```
$ node index.js
```

EXERCICE

Lister les fichiers d'un répertoire avec la commande ls -l.

/exos/05-processus/index.js

```
const exec = require('child_process').exec;
exec('ls -l', function( error, stdout) {
  if(error) { throw error };
  console.log('Command executed');
  console.log(stdout);
});
```

```
$ node index.js
```

6. GESTION DES FICHIERS

LISTER UN RÉPERTOIRE

/demos/06-files/01-list-directory/index.js

```
const fs = require('fs');

// Synchrone

console.log('--- Synchronous native mode');

const list = fs.readdirSync('./');
console.log(list);
console.log('Directory read synchronously');

// Asynchrone

console.log('--- Asynchronous native mode');

fs.readdir('./', (error, files) => {
  console.log(files);
```

```
$ node index.js
```

LIRE UN FICHIER

/demos/06-files/02-read-file/index.js

```
const fs = require('fs');

// Synchrone

console.log('--- Synchronous native mode');

let content = fs.readFileSync('./public/contacts-triptyk.txt', 'U

console.log(content);

// Asynchrone

console.log('--- Asynchronous native mode');

fs.readFile('./public/contacts-triptyk.txt', (error, file) => {
  console.log(file);
```

```
$ node index.js
```

ECRIRE DANS UN FICHIER

/demos/06-files/03-write-file/index.js

```
const fs = require('fs');

const names = ['Pierre', 'Paul', 'Jacques'];
let output = '';

names.forEach( (name) => {
  output += `${name}\n`;
});

fs.writeFile('./public/contacts.txt', output, (error, file) => {
  console.log('File created');
});

let appended = 'Appended content';

fs.appendFile('./public/contacts.txt', output, (error, file) => {
```

```
$ node index.js
```

CRÉER UN RÉPERTOIRE

/demos/06-files/04-create-directory/index.js

```
const fs = require('fs');

if(fs.existsSync('public')) { // Sync
  fs.mkdir('public', (error) => {
    if(error) throw error;
    console.log('Directory created');
  });
}
else {
  console.log('Directory public cannot be created : a directory w
}
```

```
$ node index.js
```

RENOMMER/DÉPLACER ET SUPPRIMER UN FICHIER

RENOMMER UN FICHIER

/demos/06-files/05-rename-move-delete-files/rename.js

```
const fs = require('fs');

if(fs.existsSync('public/named-file.txt')) {
  fs.renameSync('public/named-file.txt', 'public/renamed-file.txt
}
else {
  console.log('File public/named-file not found');
}
```

```
$ node rename.js
```

DÉPLACER UN FICHIER

/demos/06-files/05-rename-move-delete-files/move.js

```
const fs = require('fs');

if(fs.existsSync('./public/renamed-file.txt')) {
  fs.renameSync('./public/renamed-file.txt', './');
}
else {
  console.log('File ./public/renamed-file not found');
}
```

```
$ node move.js
```

SUPPRIMER UN FICHIER

/demos/06-files/05-rename-move-delete-files/delete.js

```
const fs = require('fs');
try {
  fs.unlinkSync('./renamed-file.txt');
}
catch(e) {
  console.log(e.message);
}
```

```
$ node delete.js
```

RENOMMER/DÉPLACER ET SUPPRIMER UN RÉPERTOIRE

RENOMMER UN RÉPERTOIRE

/demos/06-files/06-rename-move-delete-directories/rename.js

```
const fs = require('fs');
if(fs.existsSync('./public/to-delete')) {
  fs.renameSync('./public/to-delete', './public/renamed-directory
}
else {
  console.log('Directory ./public/to-delete not found');
}
```

```
$ node rename.js
```

DÉPLACER UN RÉPERTOIRE

/demos/06-files/06-rename-move-delete-directories/move.js

```
const fs = require('fs');
if(fs.existsSync('./public/renamed-directory')) {
  fs.renameSync('./public/renamed-directory', './');
}
else {
  console.log('Directory ./public/renamed-directory not found');
}
```

```
$ node move.js
```

SUPPRIMER UN RÉPERTOIRE

/demos/06-files/06-rename-move-delete-directories/delete.js

```
const fs = require('fs');

try {
  fs.rmdirSync('./renamed-directory');
}
catch(e) {
  console.log(e.message);
}
```

```
$ node delete.js
```

EXERCICES

Lister et lire les fichiers d'un répertoire

/exos/06-files/01-list-read-files/index.js

```
const fs = require('fs');
try {
  fs.readdir('./public', (error, files) => {
    if(error) throw error;
    files.forEach( (path) => {
      console.log(`We read the file ./public/${path}`);
      console.log(fs.statSync(`./public/${path}`));
      fs.readFile(`./public/${path}`, 'UTF-8', (err, file) => {
        if(err) throw err;
        console.log(file);
      });
    });
  });
}
catch(e) {
  console.log(e.message);
```

```
$ node index.js
```

Considérant les répertoires fournis pour l'exercice, vous devez effacer les répertoires et les fichiers vides. Vous devez utiliser une fonction récursive pour parser les répertoires, et vous devez bien évidemment préserver le fichier index.js à la racine.

Level up : logger les entrées supprimées dans un fichier texte placé dans ./logs/{timestamp}.txt

/exos/06-files/02-remove-files-directories/helpers/directory-helper.js

```
const fs = require('fs');
/**
* Get the text path of a resource
*
* @param {String} path
* @returns {String} as path of the directory
*/
exports.getDirectoryPath = function(path) {
  let array = path.split('/');
  return array.splice(0, array.length -1).join('/');
};

/**
* Say if a directory contains files
*
* @param {String} path
```

```
$ node index.js
```

/exos/06-files/02-remove-files-directories/modules/garbage-collector.js

```
const fs = require('fs');
const helper = require(`${process.cwd()}/helpers/directory-helper

/**
* Remove empty files/directories
*
* @param {String} path
*/
garbageCollector = function(path) {

  // Parse current path
  let items = fs.readdirSync(path);

  // Retrieve elements of current path
  items.forEach( (item) => {
```

```
$ node index.js
```

/exos/06-files/02-remove-files-directories/index.js

```
const fs = require('fs');
const GarbageCollector = require(`${process.cwd()}/modules/garbag
GarbageCollector('./');
```

```
$ node index.js
```

7. GESTION DES STREAMS

Un stream est un flux de données séquencé, éventuellement fragmenté en paquets.

https://fr.wikipedia.org/wiki/Stream

LECTURE

/exos/07-streams/index.js

```
const fs = require('fs');

let stream = fs.createReadStream('public/lorem.txt', 'UTF-8');

stream.once('data', () => {
  console.log('Start stream reading');
});

stream.on('data', (chunk) => {
  process.stdout.write(`chunck : ${chunk.length}`);
});

stream.on('end', () => {
  console.log('End stream reading');
});
```

```
$ node index.js
```

8. GESTION SERVEUR HTTP

CONNEXIONS HTTP(S)

HTTP - HTTPS

/demos/08-http/01-http/index.js

```
const Http = require('http'); // https
const options = {
  hostname : 'technocite.be',
  path: '/index.php/fr/component/detailsform/?form=1192',
  port : 443,
  method: 'GET'
};
```

```
$ node index.js
```

CRÉER UNE REQUÊTE HTTP

/demos/08-http/02-server/index.js

```
const Http = require('http');
const options = {
  hostname : 'www.triptyk.eu',
  path: '/fr/realisations',
  port : 80,
  method: 'GET'
};

let request = Http.request(options, (response) => {

  let output = '';

  console.log('Connexion');
  console.log(response.statusCode);
  console.log(response.headers);
```

```
$ node index.js
```

CONSTRUIRE LE SERVEUR

/demos/08-http/03-server/index.js

```
const Http = require('http');

const server = Http.createServer( (request, response) => {
  response.writeHead(200, { 'Content-Type' : 'text/plain' });
  response.end('Node server created');
});

server.listen(8000);

console.log('Node server listen port 8000 : http://localhost:8000
```

```
$ node index.js
```

/demos/08-http/03-server/index.js

```
const server = Http.createServer( (request, response) => {
  let output = '';
  output += `YOUR_HTML`;

  response.writeHead(200, { 'Content-Type' : 'text/html' });
  response.end('Node server created');
});

server.listen(8000);

console.log('Node server listen port 8000 : http://localhost:8000
```

```
$ node index.js
```

EXERCICE

/exos/08-http/01-http/index.js

```
const server = Http.createServer( (request, response) => {
  let output = '';
  output += `YOUR_HTML`;

  response.writeHead(200, { 'Content-Type' : 'text/html' });
  response.end('Node server created');
});

server.listen(8000);
console.log('Node server listen port 8000 : http://localhost:8000
```

```
$ node index.js
```

SERVIR DES FICHIERS STATIQUES

/demos/08-http/03-serve-static-file/index.js

```
const Http = require('http');
const fs = require('fs');
const Path = require('path');

const server = Http.createServer( (req, response) => {

  console.log(`${req.method} : request for ${req.url}`);

  if(req.url === '/')
  {
    fs.readFile('./public/templates/index.html', 'UTF-8', (error,
      response.writeHead(200, { 'Content-Type' : 'text/html' });
      response.end(output);
    });
  }
  else if(req.url === '/about.html')
```

```
$ node index.js
```

SERVIR DU JSON

RÉPONSE SERVEUR

/demos/08-http/04-serve-json-file/01-without-treatment/index.js

```
const Http = require('http');

const server = Http.createServer( (req, response) => {

  console.log(`${req.method} : request for ${req.url}`);

  const data = require('./public/files/list.json');

  if(req.url === '/')
  {
    response.writeHead(200, { 'Content-Type' : 'text/json' });
    //response.end(data);
    response.end(JSON.stringify(data));
  }
  else if(req.url === '/is-active')
  {
```

```
$ node index.js
```

RÉPONSE ET TRAITEMENT

/demos/08-http/04-serve-json-file/02-with-treatment/index.js

```
const Http = require('http');
const fs = require('fs');
const Path = require('path');

const server = Http.createServer( (req, response) => {

  console.log(`${req.method} : request for ${req.url}`);

  if(req.url === '/')
  {
    fs.readFile('./public/templates/index.html', 'UTF-8', (error,
      response.writeHead(200, { 'Content-Type' : 'text/html' });
      response.end(output);
    });
  }
  else if(req.url.match(/.css$/))
```

```
$ node index.js
```

RÉCUPÉRER DES DONNÉES EN POST

/demos/08-http/05-retrieve-post-data/index.js

```
const Http = require('http');
const fs = require('fs');
const Path = require('path');

const server = Http.createServer( (req, response) => {

  console.log(`${req.method} : request for ${req.url}`);

  if(req.method === 'GET')
  {
    fs.readFile('./public/templates/index.html', 'UTF-8', (error,
      response.writeHead(200, { 'Content-Type' : 'text/html' });
      response.end(output);
    });
  }
  else if(req.method === 'POST')
```

```
$ node index.js
```

UTILISER LE PATTERN MVC

/demos/08-http/06-mvc-server/app.js

```
const router = require('./app.router')
const http = require('http')
http.createServer( router ).listen('8001', ( error ) => {
    if(error) throw error
    console.log('Server is running')
} )
```

/demos/08-http/06-mvc-server/app.router.js

```
const routes =
[
  { url : '/', controller : 'index' }
]

module.exports = ( request, response ) => {
  let index = routes.findIndex( ( item ) => item.url === request.
  if(index !== -1)
  {
    require(process.cwd() + '/controllers/' + routes[index].contr
  }
  else
  {
    require(process.cwd() + '/controllers/error.js')(request, res
  }
};
```

/demos/08-http/06-mvc-server/controllers/index.js

```
const fs = require('fs');
const promisify = require('es6-promisify').promisify;
const read = promisify(fs.readFile);

module.exports = async ( request, response ) => {
    read( `${process.cwd()}/views/index.html`, { encoding : 'utf-
        .then( result => {
            read( `${process.cwd()}/models/contacts.json`, { enco
                let mod = JSON.parse(model);
                let template = '';
                mod.forEach( (item) => {
                    template += '
• ' + item.name + '
\r';
                });
                let output = result.replace(/{{LIST}}/, template)
```

9. OUTILS INTÉRESSANTS

NVM

Gestion des versions Node.js

https://github.com/creationix/nvm

NODEMON

Gère les changements et redémarre le serveur

https://nodemon.io/

PM2

Gestion de serveur node en production

http://pm2.keymetrics.io/

EXTENSIONS VS CODE

- Javascript ES6 code snippets
- Path intelliSense
- DotENV
- jsHint
- ESLint
- TODO Highlight
- Git history
- Auto Rename Tag

10. RESSOURCES

- https://www.nodebeginner.org/
- https://x-team.com/nodejs-resources/
- http://book.mixu.net/node/
- https://risingstack.com/resources/node-hero

11. TP FINAL

Vous développez la première étape d'un mini-site plaquette pour une agence de voyage.

Vous devez respecter le cahier des charges suivant.

CONTENUS

- Page "Home" : 1 titre, 1 description, 1 section "Top destinations" où vous affichez les 3 destinations qui ont le plus de vues.
- Page "About" : 1 titre et 1 description
- Page "Destinations" : 1 titre, 1 description et une liste de résumé des destinations (1 destination = prix, lieu, photo)
- Page "Contact" : 1 formulaire de contact (nom, email, message)

FEATURES

- Envoi du formulaire de contact avec Nodemailer
- Log des envois de mails dans un fichier ./logs/emails.txt
- Mode "Demo" qui bloque l'envoi du formulaire

CONTRAINTES TECHNIQUES

- Implémentation MVC
- Data source = fichier .json
- Ecriture du log est déclenchée par un événement custom
- Le mode "Demo" doit être activé au lancement via un paramètre
- Pas de template, vous affichez les données brutes
- Le formulaire est envoyé via une requête HTTP déclenchée par un module JS côté client.

DÉLAIS

Le client attend une première version du projet dans 8h.